Machine Learning

Priit Järv, TalTech, 2020

Overview

- What’s machine learning?
- Learning to act
- Learning unknown things

Machine learning definition

Approximate unknown function:

$$h(x) \approx f(x)$$

Boring, but true

Approximating unknown functions

Example:

x	f(x)
1	1
2	4
2.5	6.25
3	9

Approximating unknown functions

Example:

x	f(x)
1	1
2	4
2.5	6.25
3	9

Suppose you don't know how to compute f(x)

Look at the table and guess f(2.4)

Approximating unknown functions

Example:

x	f(x)
1	1
2	4
2.5	6.25
3	9

An algorithm:

1. Find closest x
2. Return f(x) from table

So…. h(2.4) = 6.25

Approximating unknown functions

Example:

x	f(x)
1	1
2	4
2.5	6.25
3	9

An algorithm:

1. Find closest x
2. Return f(x) from table

This is *totally* machine learning!

(nearest neighbors algorithm)

Approximating unknown functions

In this example, you really don’t know direct formula for f(x)

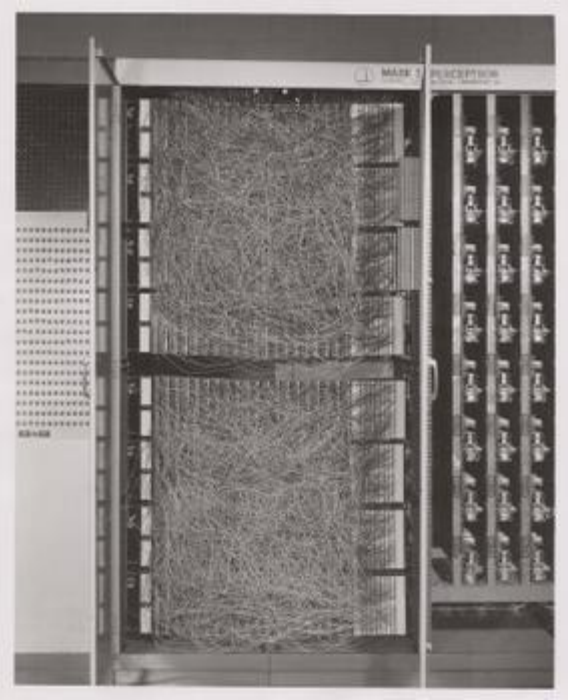
x		
f(x)	“cat”	“dog”

Approximating unknown functions

Side note: how to represent this mathematically?

Some ML ideas

Try to make something like the brain



The “Perceptron”, US Navy 1958

(early neural network ideas)

Some ML ideas

Try to make machines do stuff that seems really hard

Like play chess/checkers

Temporal difference learning, Samuel 1959

(properly solved around 2017 by DeepMind)

Some ML ideas

People did machine learning before they knew it:

The “Iris flower dataset”

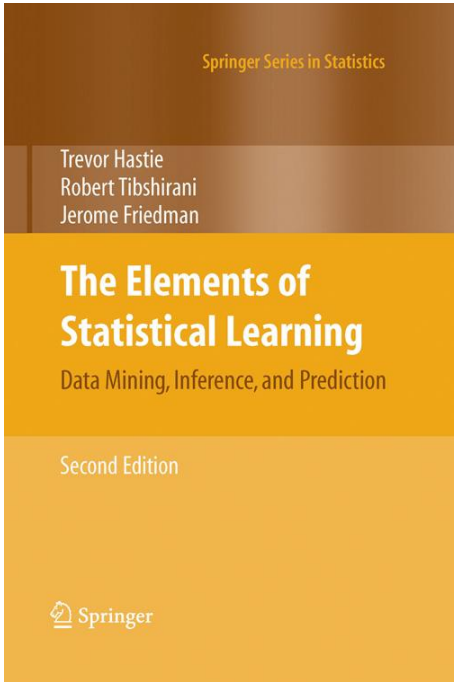
Ronald Fisher, 1936

Use **statistics** to determine species

The statistics connection

ML has strong background in statistics

In UC Berkeley, the intro course to ML uses this book:



Hint: it’s not easy bedtime reading

Let’s solve the cat/dog problem

Nearest neighbors not so great:

Similarity of images bit tricky to compute

sim( , ) = ?

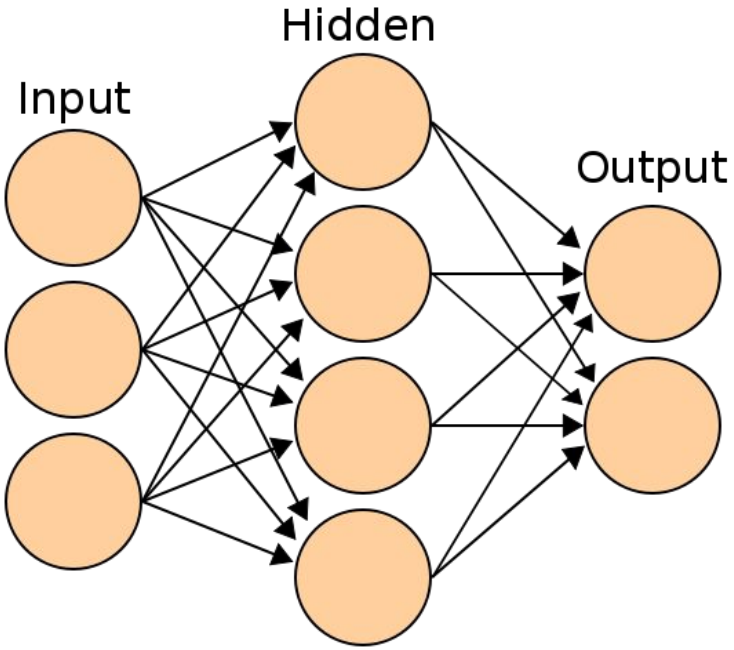
We’ll use neural networks

(we’ve heard they’re pretty good)

Neural Network

Approximation of nonlinear functions

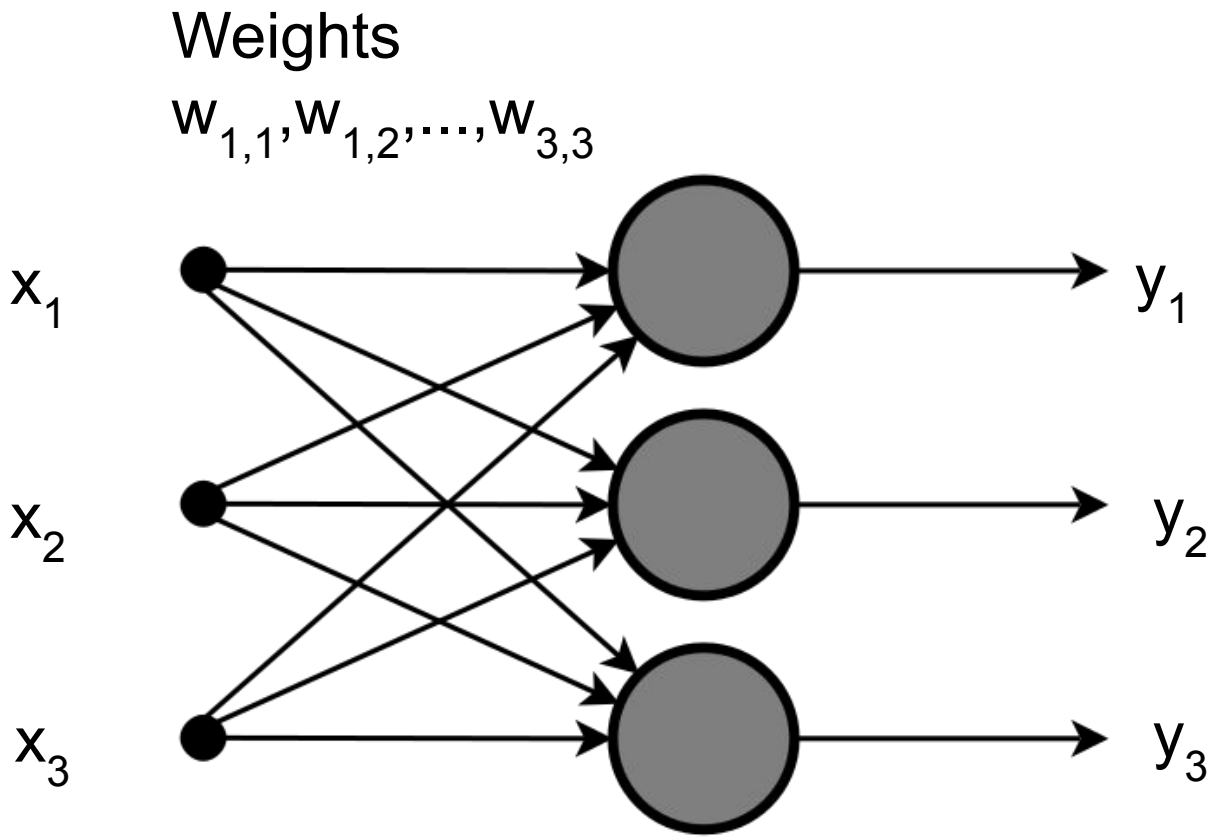
Pixels:



Cat? = 1

Dog? = 0

Neural network: single layer



Neural network: single layer

We want to minimize error on each output y_k

Inputs $x_1 ... x_n$	y_k	Desired output y	Error
Cat pixels	0.78	1	0.22
Dog pixels	0.33	0	-0.33
Another cat	0.96	1	0.04

Usually the error is squared (and called the loss function).

Neural network: single layer

Output of one unit

$$y_k = g(\sum w_{i,k} x_i)$$

$g()$ - activation function

For one training sample, input and desired output are fixed.

So error is a function of the weights

$$\boldsymbol{Err}_{\mathbf{k}} = (\text{Desired } y_k - g(\sum \boldsymbol{w}_{\mathbf{i,k}} x_i))^2$$

Neural network: single layer

Let that sink in...

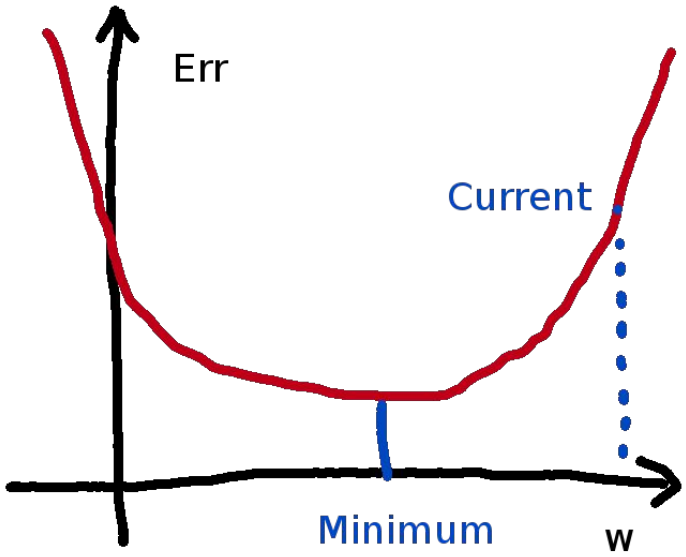
This is the key to training the network:

error is a function of the weights

$$\boldsymbol{Err}_{\mathbf{k}} = (\text{Desired } y_k - g(\sum \boldsymbol{w}_{\mathbf{i,k}} x_i))^2$$

Neural network: weight update

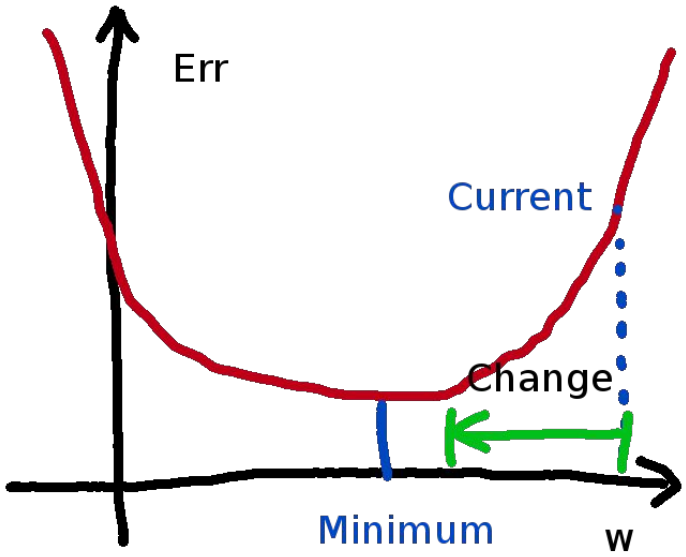
For one training sample, one output and one weight (treat other weights as fixed):



Change the weight to decrease error.

Neural network: weight update

Derivative *Err'*(*w*) gives the direction of growth, change *w* in opposite direction.



Neural network: weight update

For learning we need many **labeled examples**

A.k.a. Training data

Input vector **x**	y_0	y_1
Cat picture 1	1	0
...	...	...
Cat picture 32422	1	0
Dog picture 1	0	1

Learning to act

Can we learn behavior from examples?

- Chess board: 10^{120} possible inputs
- Helicopter in flight: also very many (depends on representation)

We **don’t know** the “right action” for each possible input

Learning to act

Reinforcement learning: try something, see if good or bad stuff happens

Robot in the video learns the behaviour of crawling forward:

https://youtu.be/f2nIKFMyfSg?t=335

Q-learning: states and actions

Crawling robot in the video

- 36 states for the arm (6 angles for each joint)
- 4 actions (two servos, two directions)

Q-learning: states and actions

Table for choosing actions (with some learned values):

Q(s,a)	a_1	a_2	a_3	a_4
s_1	10	9	-5	0
s_2	0	0	-10	0
…	…	…	…	…
s_{36}	0	0	0	3

In state *s*, choose action *a* with highest Q(*s*, *a*)

Q-learning: states and actions

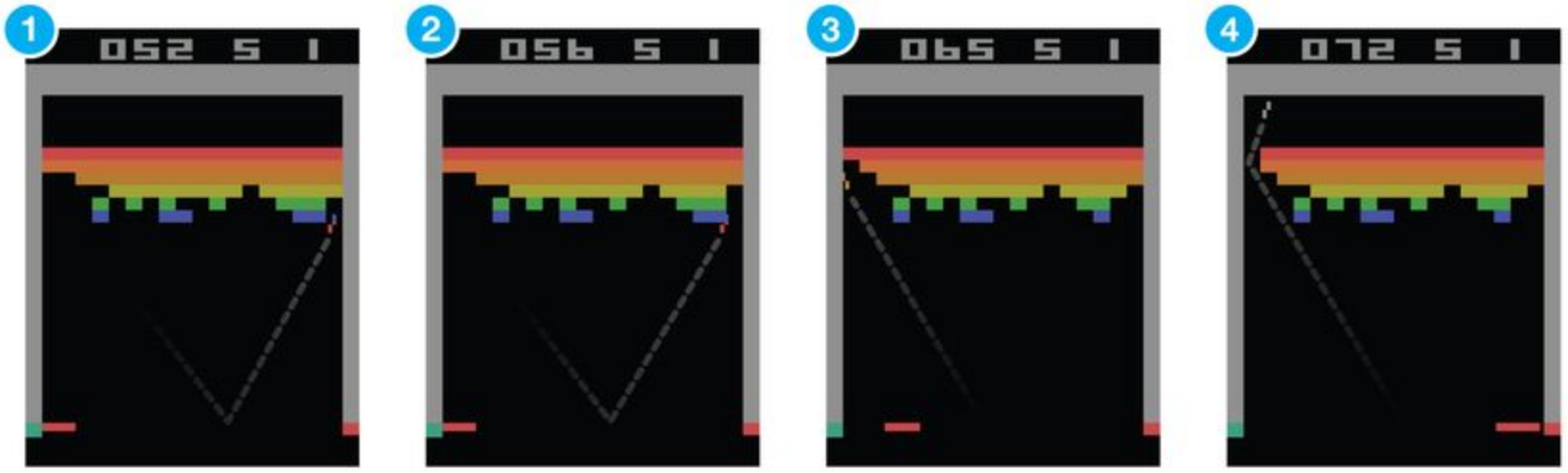
Table for choosing actions (with some learned values):

Q(s,a)	a_1	a_2	a_3	a_4
s_1	10	9	-5	0
s_2	0	0	-10	0
…	…	…	…	…
s_{36}	0	0	0	3

Choose random actions sometimes to explore

Q-learning

Can we make a table for these states?

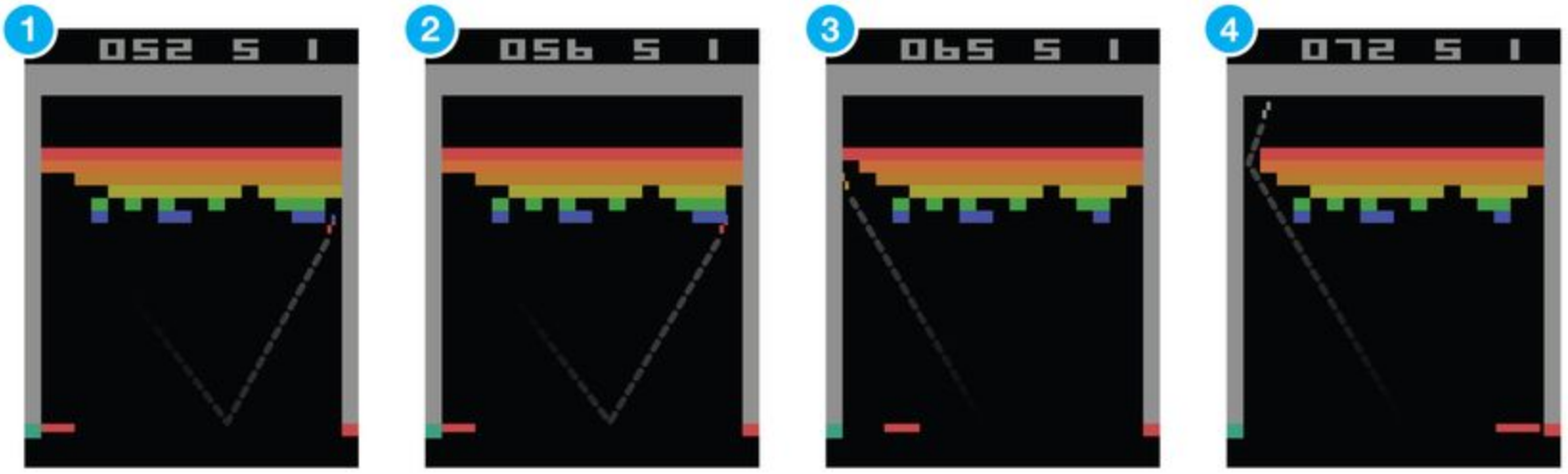


Atari game “Breakout”

(image credit: DeepMind)

Q-learning

Can we make a table for these states?



The trick is to approximate Q(*s*,*a*) with a neural network!

(Deep reinforcement learning)

Reinforcement learning

Try it yourself

https://gym.openai.com/

Good intro from University of Tartu

https://neuro.cs.ut.ee/demystifying-deep-reinforcement-learning/

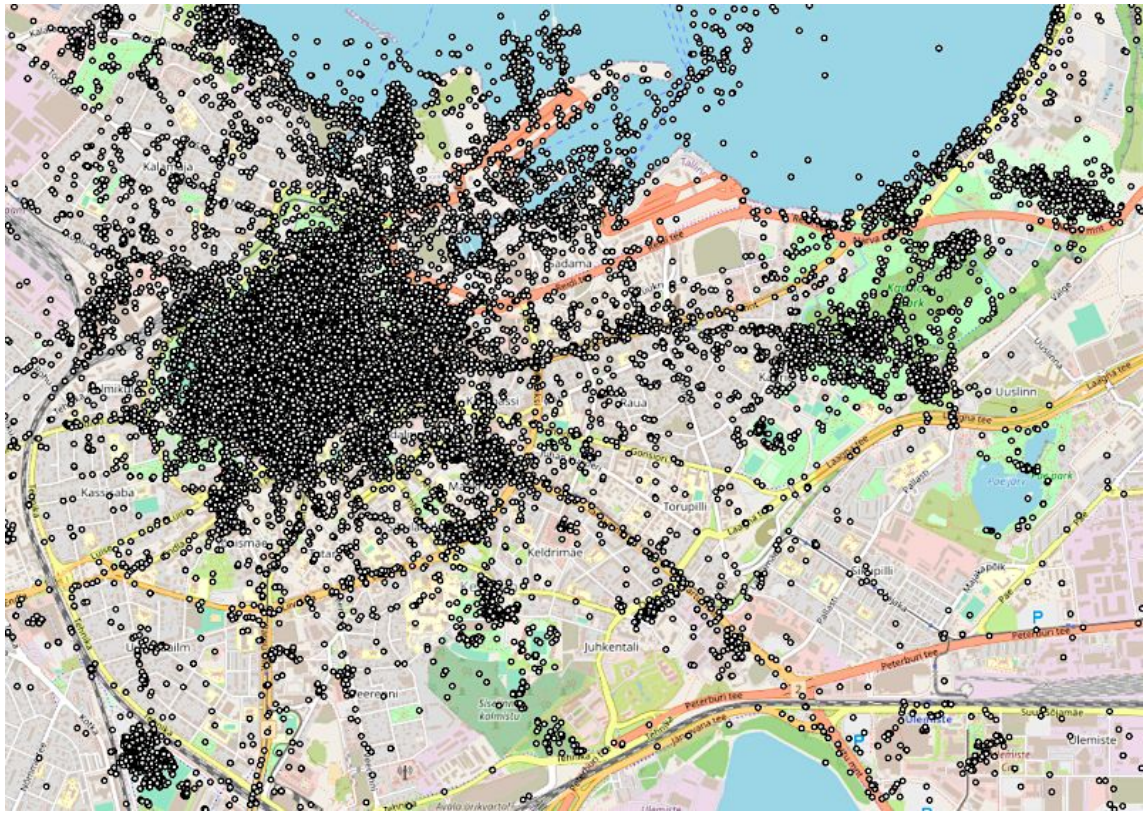
Learning unknown things

Also known as **unsupervised learning**

- No training data
- We don’t know what the result is supposed to be

Unsupervised learning

Example: clustering



data:

Flickr tourist photos

Unsupervised learning

Analysis of tourist photos

Each photo is a tiny recommendation:

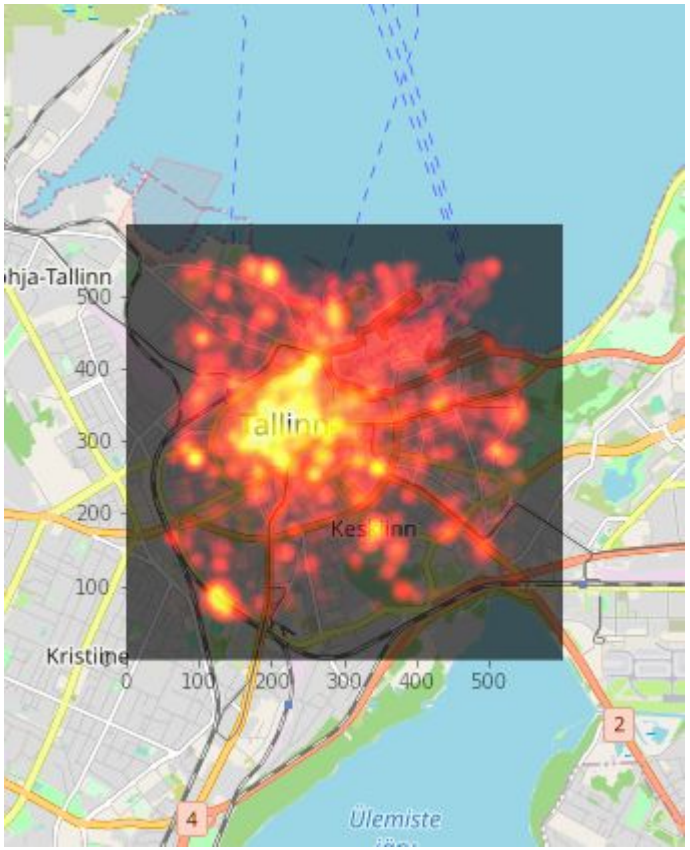
“Look, I went to see this thing, it’s interesting”

Task: use these photos to find interesting areas

Unsupervised learning

This task is about local density

(area borders are where photo density drops)



Step 1: kernel density estimation

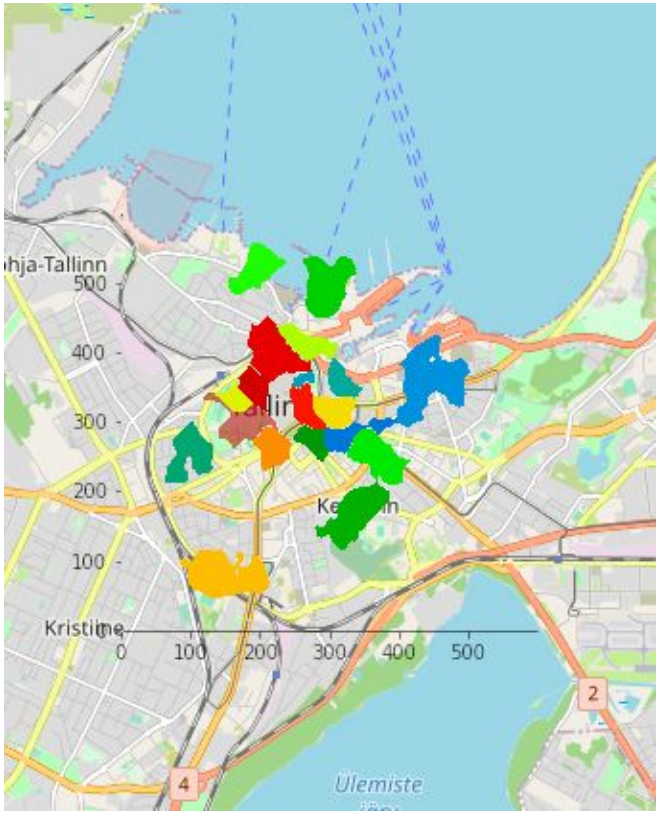
(Using algorithm from Laptev et al. “Parameter-free discovery and recommendation of areas-of-interest.”, SIGSPATIAL 2014)

Unsupervised learning

This task is about local density

(area borders are where photo density drops)

Step 2: area segmentation using change in density



Summary

- Unknown or difficult functions
- Supervised learning
- Reinforcement learning
- Unsupervised learning

Courses @TalTech

ITI0210 - intro to AI, has some ML

ITI0217 - data mining

ITB8804 - practical machine learning, recommended for you

Tons of stuff in master’s study program (IAPM)

Don’t forget to study math!